Crowd Simulation by Deep Reinforcement Learning

Jaedong Lee
Seoul National University
jaedong@mrl.snu.ac.kr

Jungdam Won
Seoul National University
jungdam@mrl.snu.ac.kr

Jehee Lee
Seoul National University
jehee@mrl.snu.ac.kr

Figure 1: A sequence of screenshots where two groups of agents colored in blue and red are simulated by a policy learned by our deep reinforcement learning approach. The two different groups pass without colliding with each other (in an order of top-left, top-right, bottom-left, and bottom-right).

ABSTRACT

Simulating believable virtual crowds has been an important research topic in many research fields such as industry films, computer games, urban engineering, and behavioral science. One of the key capabilities agents should have is navigation, which is reaching goals without colliding with other agents or obstacles. The key challenge here is that the environment changes dynamically, where the current decision of an agent can largely affect the state of other agents as well as the agent in the future. Recently, reinforcement learning with deep neural networks has shown remarkable results in sequential decision-making problems. With the power of convolution neural networks, elaborate control with visual sensory inputs has also become possible. In this paper, we present an agent-based deep reinforcement learning approach for navigation, where only a simple reward function enables agents to navigate in various complex scenarios. Our method is also able to do that with a single unified policy for every scenario, where the scenario-specific parameter tuning is unnecessary. We will show the effectiveness of our method through a variety of scenarios and settings.

CCS CONCEPTS

•Computing methodologies → Animation; Crowd simulation; Reinforcement learning;

KEYWORDS

animation, crowd simulation, collision avoidance, reinforcement learning

Motion, Interaction and Games, Limassol, Cyprus
 978-1-4503-5817-0/18/08…$15.00
DOI: 10.1145/3230744.3230782

ACM Reference format:
Jaedong Lee, Jungdam Won, and Jehee Lee. 2018. Crowd Simulation by Deep Reinforcement Learning. In *Proceedings of Motion, Interaction and Games, Limassol, Cyprus, November 8-10, 2018,* 7 pages.
DOI: 10.1145/3230744.3230782

1 INTRODUCTION

Simulating believable behaviors of virtual agents in crowded scenes has been an important research topic in not only computer graphics such as industry films, computer games, and virtual reality but also other fields such as urban engineering, emergency simulation, and behavioral science. One of the key capabilities agents should have is navigation, which is reaching goals without colliding with other agents or obstacles. The primary challenge here is that the environment changes dynamically along the progress of simulation, where the current decision of an agent can largely affect the state of other agents as well as the agent in the future.

The studies for crowd simulation can be divided into two categories, one is a microscopic approach and the other is a macroscopic approach. The microscopic approach (i.e. agent-based) assumes that agents make a decision individually from their neighborhood information. The simulation is performed in three steps for each agent, collecting neighborhood information, a decision from the information, and acting the decision. On the other hand, the macroscopic approach (i.e. system-based) assumes that there exists a *know-it-all* who knows all about the scene. The simulation is performed in three steps, building one integrated dynamics for the scene in a coarse manner, solving the dynamics of the system, and adding details (local movements of agents) afterward.

The navigation performance of the macroscopic approach is more efficient than the microscopic approach, where movements of the agents are more fluid and it suffers from less congestion. This is because the macroscopic approach solves the entire system, which enables us to plan the movements from a global point of view whereas only local planning is possible in the microscopic approach. However, the macroscopic method sometimes generates unnatural movements which do not match to real humans, the movements are too smooth and lack local details.

Recently, reinforcement learning (RL) with deep neural networks (DNN) has shown remarkable results in various topics such as physics-based animation, robotics, and playing computer games. Given an agent, its environment, and the interaction method between them, RL finds an optimal policy for the agent that maximizes cumulative rewards, which tells how good the state of the agent is. This means that the policy generated by RL is able to make a decision in a way that it predicts the future although the input for the policy is an only current state. For these advantages, we can expect the movements that the agent makes based on past experience or anticipation of the future situation. And with the power of DNN, elaborate control with visual sensory inputs has also become possible.

We present deep reinforcement learning approach for navigation in crowd simulation, where only a simple reward function enables agents to navigate in various complex scenarios. The agents in our method move in a microscopic manner where each agent make a decision independently by sensing its neighbors, however, the optimality of the decision is comparable to existing methods or better in some cases. This is because the decision is generated by the optimal policy leaned by deep reinforcement learning. Without the parameter tuning for each scenario, our method can construct a single unified policy that works in many scenarios including unseen ones in the learning process. We will show the effectiveness of our method through a variety of scenarios/settings that range from simple and coarse scenes to complex and dense scenes.

2 RELATED WORK

One stream of researches for simulating believable agents in virtual environments is a microscopic approach which is an agent-based method, where each agent makes a decision individually from its neighborhood information for every time-step. Two popular decision models are force-based and velocity-based models.

The mechanism of the force-based model is similar to a physics simulation, where each agent receives virtual forces generated from the spatial or social relationship between the agent and its neighbors, then the state (velocity, position) of the agent are integrated accordingly. Reynolds [1987] presented a flocking simulation model for birds. Helbing and his colleagues [2000; 1995] proposed social force model for normal and panic situations. Pelechano et al. [2007] proposed an individual control in dense environments. Karamouzas et al. [2014] defines a time-to-collision dependent potential energy whose derivatives generate forces.

In the velocity-based model, each agent selects a velocity that minimizes a given cost function, then the position of the agent is integrated by the velocity. The velocity-based model is usually less sensitive to parameter choice and more stable in a large time-step than the force-based model because the velocity-position relationship is closer than the force-position relationship. There have been studies on velocity obstacles that is a set of collidable velocities and several variations on the method [Guy et al. 2009, 2012, 2011; van den Berg et al. 2009, 2008]. Visual sensory inputs was integrated into a velocity-based model to imitate the mechanism of real human [Ondřej et al. 2010; Vaina et al. 2010].

Another stream of researches is a macroscopic approach which is a system-based method, where states of all agents are computed from an omniscient point of view. This usually deals with the movement of entire agents as a mixture of global and local movements. A collision-free navigation is first achieved when generating the global movement, where the system dynamics is solved in a coarse manner, then the local movement is added along the computed trajectories. Approaches based on continuum theory have been studied [Hughes 2002; Treuille et al. 2006]. Narain et al. [Narain et al. 2009] extended the approach in a large scale scenario by using variational constraint called unilateral incompressibility.

Behaviors of a real human in the crowd emerge by a variety of causes that range from internal motives such as a specific purpose or emotional states to external motives such as distances to neighbors or their spatial formation. Several different causes can also affect the behaviors simultaneously. Due to the difficulty on designing such complex interactions by virtual forces or cost functions, there have been several attempts to combine real captured data with existing approaches to add realism [Ju et al. 2010; Lee et al. 2006; Lerner et al. 2007; Pettré et al. 2009; Qiao et al. 2018], for analysis on simulated results [Charalambous et al. 2014; Guy et al. 2012]. There have also been approaches on direct manipulation for crowd scenes [Kim et al. 2014; Kwon et al. 2008; Normoyle et al. 2014].

Reinforcement learning (RL) is an algorithm to solve a Markov decision process (MDP), which is a mathematical formulation on sequential decision-making problems. RL computes an optimal

policy (direct mapping from states to actions) which maximizes the expected return defined for each problem. This implies that the policy learned by RL is able to generate anticipatory behaviors, which is desirable in crowd simulation. Several approaches using deep neural networks in RL have recently been proposed, where they showed remarkable improvement for several challenging problems [Mnih et al. 2015; Silver et al. 2016]. Due to these potentials of RL (and DRL) approach several researches have adopted it in crowd simulation [Bastidas 2014; Casadiego and Pelechano 2015; de la Cruz et al. 2015; Henry et al. 2010; Martinez-Gil et al. 2012, 2015; Torrey 2010], however, simple and small scale environments were tested in those studies. Furthermore, there is no DRL approach to the best of our knowledge. And several deep learning approaches, not RL, show plausible results. Someone use regression neural network to predict characteristic of crowd and the other use RNN and GAN framework to generate realistic trajectory [Alahi et al. 2016; Gupta et al. 2018; Liu et al. 2017].

3 APPROACH

3.1 Simulation

An agent in crowd is represented as an oriented circle (p, θ) in 2D whose radius is r, where $p \in \mathrm{R}^2$ and $\theta \in \mathrm{R}$ are the position and the orientation of the agent, respectively. Obstacles are represented in a similar fashion except for that they don't have orientations. At each time-step t, the agent makes a decision (v, ω) and the simulation proceeds.

$$\begin{aligned} p_{t+1} &\leftarrow p_t + h \cdot v \\ \theta_{t+1} &\leftarrow \theta_t + h \cdot \omega \end{aligned} \tag{1}$$

where h is the size of simulation time-step. We consider that collisions are occurred between agents or between an agent and an obstacle if the distance is smaller than the sum of radius.

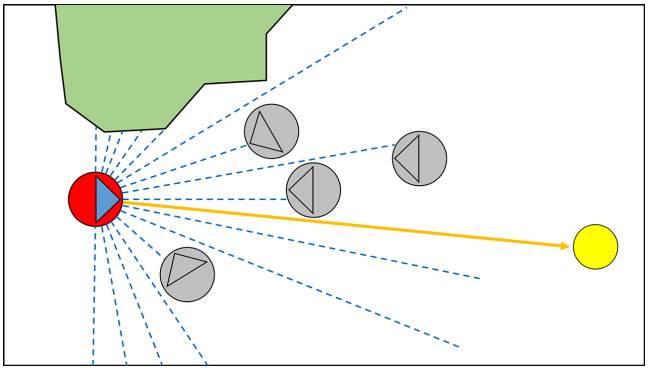

Figure 2: State representation. The state of an agent (red) is composed of a direction to the target (yellow) and visual sensory inputs (blue dots) in an environment where other agents (gray) and obstacles (green) exist.

3.2 Reinforcement Learning

Reinforcement learning (RL) solves a sequential decision-making problem, which is defined as Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions (decisions), $\mathcal{P}(s, a, s')$ is a transition probability to the next state s' given the current state s and the action a, $\mathcal{R}(s, a, s')$ is a reward function which tells the desirability of the agent state, and $\gamma \in [0, 1)$ is a discount factor which prevents the sum of rewards from becoming infinity. RL finds an optimal policy $\pi(s)$ that maximizes the expectation on cumulative rewards $\eta(\pi)$,

$$\eta(\pi) = E_{s_0, a_0, \cdots}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big] \tag{2}$$

where $s_t \sim \mathcal{P}(s_{t-1}, a_t, s_t)$, $a_t \sim \pi(s_t)$, and $r_t = \mathcal{R}(s_{t-1}, a_t, s_t)$. The core RL is to construct an approximation on $\eta(\pi)$ called a value function. We consider a state-action value function $Q(s, a)_\pi$ gives the cumulative reward when taking a and following π afterwards. The true (optimal) value function satisfies the Bellman optimality equation.

$$Q(\mathbf{s}, \mathbf{a}) = \mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in A'} Q(\mathbf{s}', \mathbf{a}'), \tag{3}$$

where A' is all possible actions at the subsequent state $\mathbf{s}'$.

3.3 State and Action

The state $s = (s_{int}, s_{ext})$ of an agent is composed of the internal state s_{int} and the external state s_{ext} (see Figure 2). The internal state $s_{int} = M^{-1}(\bar{p} - p)$ measures the relative position of the agent computed in the local coordinate of the agent, where $M \in \mathcal{R}^{2x2}$ and p are the orientation and the positions of the agent, $\bar{p}$ is the position of the goal. The external state $s_{ext} = (D_{-K}, \cdots, D_{-1}, D_0)$ identifies neighborhood agents through visual sensors, where $D_{-i} = (d_1, \cdots, d_N)$ is an depth map measured in i-th previous time-step from the current time and d_i is a depth value measured by i-th ray. 3 depth maps (K=3) with 20 rays (N=20) for each map in the span angle of 180 degrees are used in our experiment. These parameters are set heuristically based on experiments. In the case of low number of ray, it shows low performance but it shows similar performance when the ray number increases. The effect of depth map number also shows a similar result. The action $a = (v, \omega)$ is equal to the decision of the simulation as described in Equation 1 which is composed of velocities of the position and the orientation.

3.4 Rewards

The goal of navigation is to reach a goal without colliding with other agents or any obstacles in the scene. It is also desired that the movement of the agent is smooth. We designed the reward function $\mathcal{R}$ that has three terms.

$$\mathcal{R} = r_{goal} + r_{collision} + r_{smooth} \tag{4}$$

The first term r_{goal} penalizes a large deviation of the agent from the goal.

$$r_{goal} = w_1(\bar{l} - l) \tag{5}$$

where $\bar{l}$ and l are distances to the goal from the agent at previous and current time-steps, respectively. The second term $r_{collision}$ checks agent-agent or agent-obstacle collisions, then gives a penalty if it is occurred.

$$r_{collision} = \begin{cases} -w_2 & \text{if collision occurs} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

If there exist multiple collisions for the agent, we considered it as one collision rather than accumulating them. The last term r_{smooth}

Table 1: Parameters

Simulation time step (h)	0.1
Learning rate (π)	0.0001
Learning rate (Q)	0.001
Discount factor (γ)	0.95
Exploration probability (ρ_0,ρ_1)	(0.5, 0.2)
Exploration noise (Σ)	$0.05I$
Batch size	32
Target update rate (τ)	0.001
(v_{min}, v_{max}) (m/s)	$(-0.5, 1.5)$
$(\omega_{min}, \omega_{max})$ (rad/s)	$(-\pi/4, \pi/4)$
w_1	4.0
w_2	3.0
w_3	4.0
w_4	1.0

plays an role of regularization which encourages the agent to move smoothly.

$$r_{smooth} = -w_3 FLOOD(v, v_{min}, v_{max}) - w_4 FLOOD(\omega, \omega_{min}, \omega_{max}) \tag{7}$$

where $FLOOD(x, x_{min}, x_{max}) = \|min(x - x_{min}, 0)\| + \|max(x - x_{max}, 0)\|$ measures how much x deviates from the given range (x_{min}, x_{max}).

3.5 Learning

Our formulation for crowd simulation has continuous state and action spaces. To deal with it, we utilize actor-critic method that is similar to [Lillicrap et al. 2015]. The actor (policy) and the critic (value function) are represented as deep neural networks $\pi(s|u)$ and $Q(s, a|w)$, where u and w are parameter values (weight and bias) of the networks. Given a batch of transition of the agent $\{(s_1, a_1, r_1, s_1'), \cdots, (s_M, a_M, r_M, s_M')\}$, where s_i, a_i, r_i, and s_i' are the current state, action, reward, and next state at i-th transition, the critic network Q is learned by minimizing the loss L that measures how much Q satisfies the Bellman equation.

$$L = \frac{1}{M} \sum_{i}^{M} (r_i + \gamma Q(s_i', \pi(s_i')|\bar{w}) - Q(s_i, a_i|w)) \tag{8}$$

where γ is a discount factor, $\bar{w}$ is fixed parameter of the network which is updated periodically to the current parameter to stabilize the learning. The actor network π is learned by the deterministic policy gradient ∇J.

$$\nabla J = \frac{1}{M} \sum_{i=1}^{M} \nabla_a Q(s, a|w)|_{s=s_i, a=\pi(s_i)} \nabla_u \pi(s|u)|_{s=s_i} \tag{9}$$

where ∇_a, ∇_u are partial derivatives with respect to a, u, respectively.

4 EXPERIMENTS

We implemented our system in C++ for the simulation of agents and Python for the learning a controller. TensorFlow [TensorFlow 2015] was used for deep neural network operations. Computations were run on a PC equipped with CPU (i7-6850K) and we did not utilize the power of GPU. All parameters for the simulation and reinforcement learning are summarized in Table 1. The same values are used for all experiments unless additional comments exist. For the exploration, we start from the initial exploration probability ρ_0 then linearly decrease it to ρ_1 until 10K of training tuples are collected. The exploration noise Σ corresponds to 5% of the range of action, $(-0.2, 2.0)$ and $(-1.0, 1.0)$ are used for linear and angular velocities, respectively.

The deep neural networks used in our experiments are shown in Figure 3. The states are first fed into separate layers depending on their types, where fully connected, convolutional layers with *elu* activation units are used for internal, external states, respectively. The two layers are then combined by one fully connected layer with *linear* units. The critic Q network has the same structure except for that it includes an additional fully connected layer for the action and the final outcome is a scalar value.

4.1 Scenarios

We first tested our algorithm for 4 different scenarios which are commonly used to evaluate the performance in crowd simulation (see Figure 4). A separate policy was learned for each scenario and we used an easy-to-hard learning scheme which is similar to a curriculum learning. A policy is first learned for an environment where only an agent and a target exist, then the policy is learned again for the same environment with a few obstacles or other agents. Similarly, we increase the complexity of the environment until it matches to the original scenario. Figure 5 shows learning curves for the *Obstacle* scenario, where the horizontal axis is the number of learning iteration and the vertical axis is the performance, which is measured by a fixed set of randomly perturbed environment for the scenario. Moving-average with a window size 5 was used to show smooth learning curves.

Obstacle. This scenario is the simplest one in our experiment. There exist only one agent with its corresponding target and 20-30 static obstacles whose locations are randomly generated between the agent and the target.

Hallway. Two groups of agents are located on opposite sides and other sides are blocked by walls. The positions of targets are given in the vicinity of the initial location of the opposite group, where a lot of collisions could happen in the middle.

Crossway. This scenario can be considered as a modified version of *Hallway* scenario. One group is rotated in 90 degrees, which means that the two group are located at adjacent cardinal points. Similar to *Hallway* scenario, targets are given in the opposite location (north to south, east to west).

Circle. There exist 8-24 agents whose initial position are located on the boundary of a circle with a radius of 15 meters at equidistance. Because targets are given on opposite side of the initial position, all agents will be gathered at the center of the circle.

4.2 Generalization

Our algorithm can generate a single unified policy that works in all scenarios by utilizing the power of deep representation. To learn such policy, we randomly select an environment from *the*

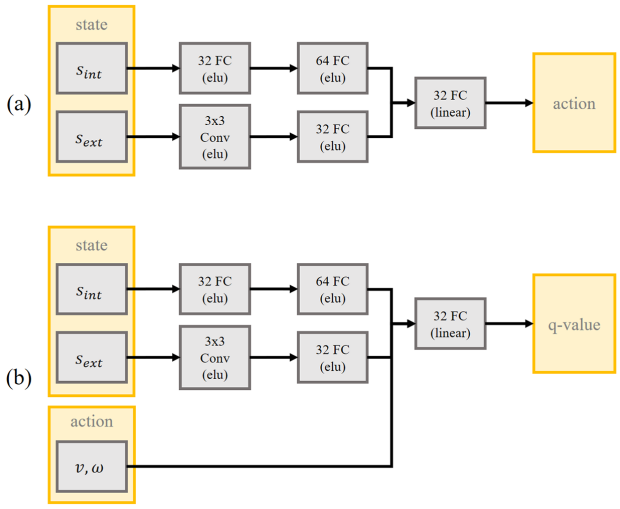

Figure 3: The structure of deep neural networks for the actor (a) and the critic (b).

environment pool during the learning, which is generated by randomly perturbing each scenario with the different level of difficulty setting. Easy environments are chosen at high probability in the early learning phase, then it decreases according to the progress of the learning. The supplemental video shows that behaviors of agents generated by the single unified policy are comparable to ones generated by the scenario-specialized policies.

Our policy decides actions from local information and it is exposed to a variety of environments during the learning. These enable us to learn a general policy that is able to be adapted to unseen scenarios as shown in Figure 6. Unseen scenarios could be generated by modifying a scenario used in the learning, combining several scenarios, or designing new scenarios from the scratch. Our method showed plausible performance in all cases to some extent.

4.3 Comparison

We compared our algorithm to two popular local planning methods *ORCA* [van den Berg et al. 2009] and *Power law* [Karamouzas et al. 2014]. *ORCA* is a velocity-based method which utilizes velocity obstacle that defines a set of collidable velocities for an agent assuming that other agents maintain their current velocities. *Power law* is a force-based method that uses a universal power law generated by analyzing real pedestrian data. For implementations for both algorithms, we utilized the versions uploaded in the authors' webpages and modified parameters on the basis of their default parameter values. Our single unified policy does not require extra parameter tuning for each scenario which was necessary for the other methods. The resultant behaviors of agents are visually comparable to other methods in most scenarios. *ORCA* has a difficulty on in *Circle* scenario, the same behaviors generated from the same state observations for all agents causes a deadlock at the end. Our method can also work well in a relatively large time-step 0.2s, a force-based method *Power Law* shows unstable agent behaviors on such setting.

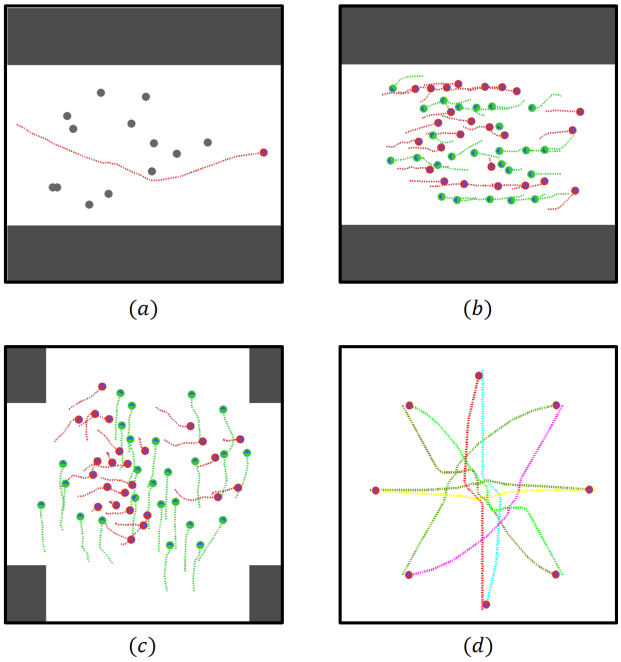

Figure 4: Scenarios. (a) Obstacle. (b) Hallway. (c) Crossway. (d) Circle.

5 CONCLUSION

We have presented an agent-based approach for simulating virtual agents in crowded environments. Our deep reinforcement learning formulation for crowd simulation was able to simulate a wide range of scenarios with only a simple reward function, where defining complex rules or tuning parameters for each scenario is unnecessary. It also enables us to learn a single unified policy that can adapt all the scenarios.

The visual sensory input used in our experiments is a set of consecutive depth maps measured by an agent. Because it has only depth values, our agents do not have a capability to distinguish other agents from obstacles, which sometimes entails unrealistic behaviors. For example, an agent should behave differently in two situations where one is being blocked by other agents and the other is being blocked by walls because other agents are dynamic entities and the wall is not. A depth map with extra labels could be incorporated into our system to solve the problem.

Although our method was able to reach different targets successfully in many scenarios, the resultant trajectories do not correspond to the shortest path/time in some cases. This is because of that r_{goal} is defined in a continuous domain and it is stationary (time independent). Assume two possible episodes where one is reaching the target in the shortest path and the episode terminates, the other is spinning around the target without termination. The continuous-stationary reward function sometimes classifies the second case as the optimal behaviors because it allows the agent to collect positive and steady reward signals even if they do not actually reach its target. One possible solution is to use a sparse (discrete) reward where agents receive it only when they actually touch the target,

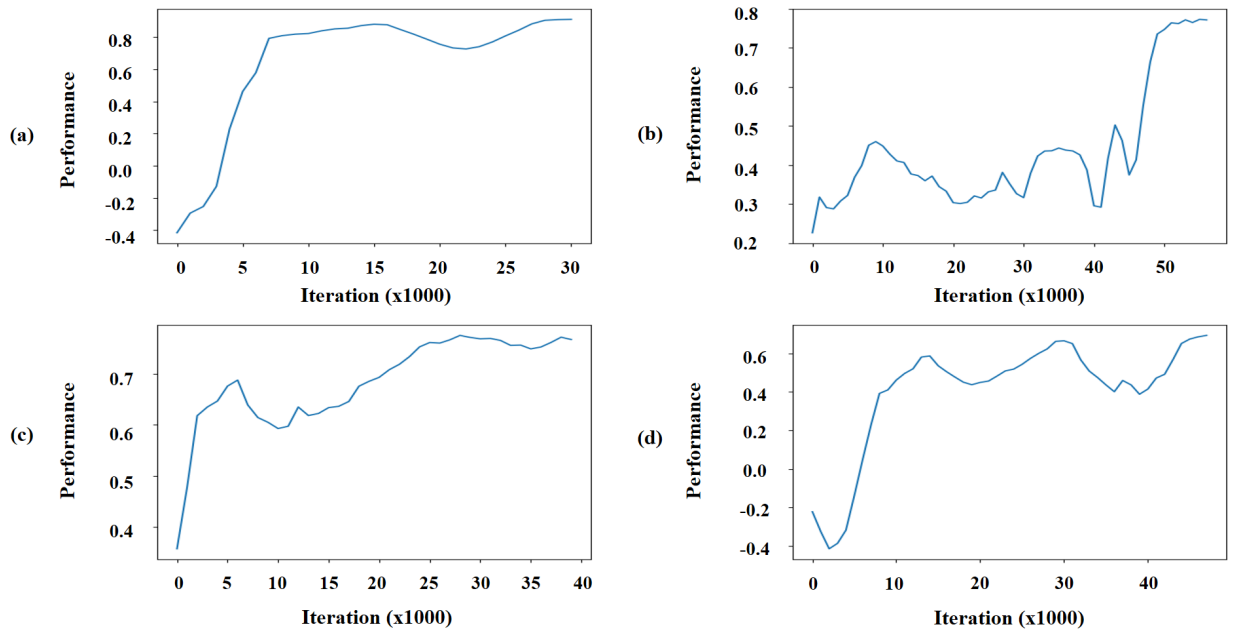

Figure 5: Learning curve. (a) Obstacle. (b) Hallway. (c) Crossway. (d) Circle.

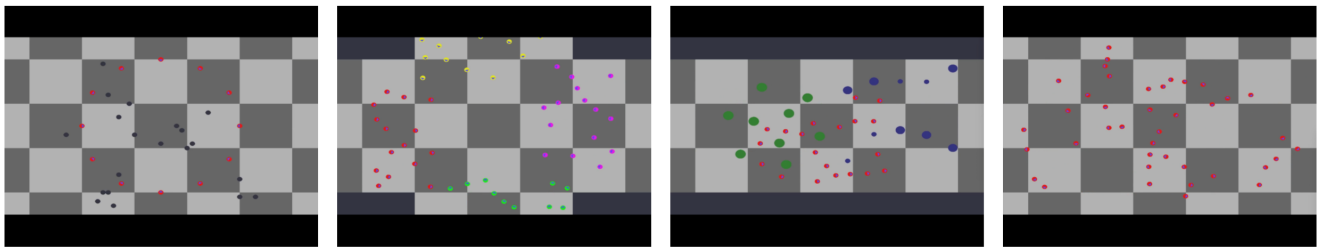

Figure 6: Generalization. (a) Circle scenario with obstacles. (b) Crossway scenario with doubled agents. (c) Moving obstacles. (c) Random goals.

however, it has been known that learning from a sparse reward is quite challenging in high-dimensional continuous environments such as ours.

There are many other exciting directions to explore in the future. Because our method is quite powerful and general, which gives us several benefits. For example, we can train more sophisticated agent models such as an oval-shaped, a dumbbell-like, or freeform shapes without a new mathematical or energy models should be developed for other methods. And we can design various types of agents not only about shape, but also about speed or characters using network input parameters. So in training time, we can train various type of agents with more complicated parameters. Those models, which is more human-like than a circle shape, could generate more realistic collision avoidance behaviors such as a shoulder strategy which had been reported as an important factor in realism [Hoyet et al. 2016]. We are also interested in multi-agent reinforcement learning (MRL). Because the reward function depends on several agents in MRL, collaborative behaviors could emerge in the learning, which will give another level of realism to crowd simulation.

ACKNOWLEDGMENTS

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the SW STARLab support program (IITP-2017-0536-20170040) supervised by the IITP(Institute for Information & communications Technology Promotion.

REFERENCES

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*
Luiselena Casadiego Bastidas. 2014. .

Luiselena Casadiego and Nuria Pelechano. 2015. From One to Many : Simulating Groups of Agents with RL Controllers. In *Intelligent Virtual Agents: 15th International Conference*. 119–125.

Panayiotis Charalambous, Ioannis Karamouzas, Stephen J. Guy, and Yiorgos Chrysanthou. 2014. A Data-Driven Framework for Visual Crowd Analysis. *Comput. Graph. Forum* (2014).

Gabriel V. de la Cruz, Bei Peng, Walter S. Lasecki, and Matthew E. Taylor. 2015. Towards Integrating Real-Time Crowd Advice with Reinforcement Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces Companion (IUI Companion '15)*. 17–20.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stephen J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. 2009. ClearPath: Highly parallel collision avoidance for multi-agent simulation. In *Symposium on Computer Animation 2009 - ACM SIGGRAPH / Eurographics Symposium Proceedings*.

Stephen J. Guy, Sean Curtis, Ming C. Lin, and Dinesh Manocha. 2012. Least-effort trajectories lead to emergent crowd behaviors. *Phys. Rev. E* (2012).

Stephen J. Guy, Sujeong Kim, Ming C. Lin, and Dinesh Manocha. 2011. Simulating Heterogeneous Crowd Behaviors Using Personality Trait Theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Stephen J. Guy, Jur van den Berg, Wenxi Liu, Rynson Lau, Ming C. Lin, and Dinesh Manocha. 2012. A Statistical Similarity Measure for Aggregate Crowd Dynamics. *ACM Trans. Graph.* (2012).

D. Helbing, I. Farkas, and T. Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407 (2000), 487–490.

Dirk Helbing and Péter Molnár. 1995. Social force model for pedestrian dynamics. *Phys. Rev. E* (1995).

P. Henry, C. Vollmer, B. Ferris, and D. Fox. 2010. Learning to navigate through crowded environments. In *2010 IEEE International Conference on Robotics and Automation*. 981–986.

Ludovic Hoyet, Anne-Helene Olivier, Richard Kulpa, and Julien Pettr'e. 2016. Perceptual Effect of Shoulder Motions on Crowd Animations. *ACM Trans. Graph.* 35, 4, Article 53 (2016), 53:1–53:10 pages.

Roger L. Hughes. 2002. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological* (2002).

Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. 2010. Morphable Crowds. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10)*. ACM.

Ioannis Karamouzas, Brian Skinner, and Stephen J. Guy. 2014. Universal Power Law Governing Pedestrian Interactions. *Phys. Rev. Lett.* 113 (2014). Issue 23.

Jongmin Kim, Yeongho Seol, Taesoo Kwon, and Jehee Lee. 2014. Interactive Manipulation of Large-scale Crowd Animation. *ACM Trans. Graph.* (2014).

Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Shigeo Takahashi. 2008. Group Motion Editing. *ACM Trans. Graph.* 27, 3 (2008), 80:1–80:8.

Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. 2006. Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data. *ACM Trans. Graph.* 25, 3 (2006), 898–906.

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by Example. (2007).

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

Weining Liu, Vladimir Pavlovic, Kaidong Hu, Petros Faloutsos, Sejong Yoon, and Mubbasir Kapadia. 2017. Characterizing the Relationship Between Environment Layout and Crowd Movement Using Machine Learning. In *Proceedings of the Tenth International Conference on Motion in Games (MIG '17)*. ACM, Article 2, 6 pages.

Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. 2012. Multi-agent Reinforcement Learning for Simulating Pedestrian Navigation. In *Adaptive and Learning Agents*. 54–69.

Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernndez. 2015. Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. 29 (01 2015), 98–130.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. 2009. Aggregate Dynamics for Dense Crowd Simulation. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*.

Aline Normoyle, Maxim Likhachev, and Alla Safonova. 2014. Stochastic Activity Authoring with Direct User Control. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '14)*. ACM.

Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. 2010. A Synthetic-vision Based Steering Approach for Crowd Simulation. In *ACM SIGGRAPH 2010 Papers*.

N. Pelechano, J. M. Allbeck, and N. I. Badler. 2007. Controlling Individual Agents in High-density Crowd Simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '07)*.

Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. 2009. Experiment-based Modeling, Simulation and Validation of Interactions Between Virtual Walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM.

Gang Qiao, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. 2018. The Role of Data-Driven Priors in Multi-Agent Crowd Trajectory Estimation. (2018).

Craig W. Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Comput. Graph.* 21 (1987).

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (2016), 484–489.

TensorFlow. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

Lisa Torrey. 2010. Crowd Simulation via Multi-agent Reinforcement Learning. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'10)*. 89–94.

Adrien Treuille, Seth Cooper, and Zoran Popović. 2006. Continuum Crowds. *ACM Trans. Graph.* (2006).

L. M. Vaina, Scott A. Beardsley, and Simon K. Rushton. 2010. *Optic Flow and Beyond* (1st ed.). Springer Publishing Company, Incorporated.

Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. 2009. Reciprocal n-body Collision Avoidance. In *INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH*.

Jur P. van den Berg, Ming C. Lin, and Dinesh Manocha. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. *2008 IEEE International Conference on Robotics and Automation* (2008), 1928–1935.